Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 40 (2014) 143 – 150

Procedia Computer Science

Fourth International Conference on Selected Topics in Mobile & Wireless Networking (MoWNet'2014)

Service provisioning through opportunistic computing in mobile clouds

Davide Mascitti[a], Marco Conti[a], Andrea Passarella[a], Laura Ricci[b]

[a]*IIT-CNR, Via G. Moruzzi 1, 56124, Pisa, Italy*
[b]*Department of Computer Science, University of Pisa, Largo B. Pontecorvo 3, 56127, Pisa, Italy*

Abstract

Mobile clouds are a new paradigm enabling mobile users to access the heterogeneous services present in a pervasive mobile environment together with the rich service offers of the cloud infrastructures. In mobile computing environments mobile devices can also act as service providers, using approaches conceptually similar to service-oriented models. Many approaches implement service provisioning between mobile devices with the intervention of cloud-based handlers, with mobility playing a disruptive role to the functionality offered by of the system. In our approach, we exploit the opportunistic computing model, whereby mobile devices exploit direct contacts to provide services to each other, without necessarily go through conventional cloud services residing in the Internet. Conventional cloud services are therefore complemented by a mobile cloud formed directly by the mobile devices. This paper exploits an algorithm for service selection and composition in this type of mobile cloud environments able to estimate the execution time of a service composition. The model enables the system to produce an estimate of the execution time of the alternative compositions that can be exploited to solve a user's request and then choose the best one among them. We compare the performance of our algorithm with alternative strategies, showing its superior performance from a number of standpoints. In particular, we show how our algorithm can manage a higher load of requests without causing instability in the system conversely to the other strategies. When the load of requests is manageable for all strategies, our algorithm can achieve up to 75% less time spent in average to solve requests.

Peer-review under responsibility of organizing committee of Fourth International Conference on Selected Topics in Mobile & Wireless Networking (MoWNet'2014)
Keywords: mobile cloud computing; opportunistic networks; mobility; service composition; analytical modelling

1. Introduction

In the last years, mobile personal devices became more and more widespread, following a trend that sees an even greater diffusion in the future[1]. Exploiting the rich networking capabilities of mobile personal devices, it is also possible to establish self-organising networks among them, such that they can communicate directly with each other. This

E-mail addresses: davide.mascitti@iit.cnr.it (Davide Mascitti)., marco.conti@iit.cnr.it (Marco Conti)., andrea.passarella@iit.cnr.it (Andrea Passarella)., ricci@di.unipi.it (Laura Ricci).

Peer-review under responsibility of organizing committee of Fourth International Conference on Selected Topics in Mobile & Wireless Networking (MoWNet'2014)
doi:10.1016/j.procs.2014.10.042

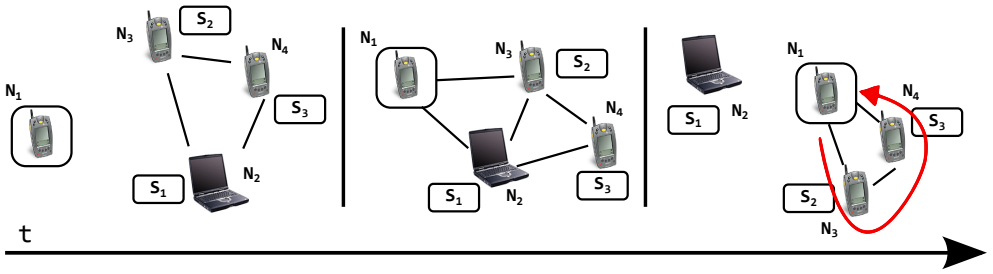


Figure 1: Service composition: a possible scenario

paradigm, studied extensively in the last decade, is nowadays gaining additional momentum, thanks to the interest into building mobile clouds, and to the possibility to exploit not only networking, but more generally rich computing capabilities of mobile devices.

Many mobile cloud systems rely on the fixed proximate or remote cloud infrastructure to access services and manage interactions between the mobile users[2]. A complementary approach considered in the mobile cloud domain[3] consists, instead, in enabling service provisioning between mobile devices directly, exploiting self-organising networking to enable direct communication between them. According to this approach, mobile nodes form a mobile cloud among them, that complements conventional cloud services available through the Internet. This type of mobile cloud can work without infrastructure support, neither at the network level nor at the service level. Therefore, it also contributes to avoid network capacity crunches that are being experienced in broadband wireless (e.g., cellular) networks due to the dramatic surge of mobile data traffic demand[1]. In addition, direct service provisioning between nearby mobile devices can implicitly exploit contextual information on physical co-location of users, that can be cumbersome to reconstruct in more conventional cloud architectures.

This type of mobile clouds can be implemented as self-organizing mobile networks as in the opportunistic networks paradigm[4], where direct contacts between mobile nodes are exploited to transfer messages without requiring stable multi-hop end-to-end paths between source and destinations. This type of mobile cloud is therefore an instance of the opportunistic computing paradigm[5], where any resource (abstracted as a service), that is available on devices in the environment can be exploited opportunistically when devices come into direct contact.

In opportunistic computing, services provided shared by different devices can be composed dynamically to provide higher-level functionality that is not available on individual devices alone. However, given a particular service, there can be multiple ways of composing resources, and more devices providing the required components. Therefore, opportunistic computing solutions must be able to select the best alternative among the available ones, taking into account the fact that nodes providing service components may be only intermittently connected, because of the nature of the underlying mobile environment.

In Figure 1 we see a possible example scenario. A user with his phone (in the following, the *seeker*, marked as node N_1), needs a service which is not available on the local device. This service, marked as S_1, can be provided by node N_2, but a viable alternative comes also from the sequential composition of services S_2, S_3, provided respectively by nodes N_3 and N_4. As N_1 comes in contact with node N_2, it also comes in contact with node N_3 that can serve as the starting point for the composition S_2, S_3. By analysing the relationship among the devices, N_1 can realise that N_2 will not remain in connection range until the completion of S_1 and the download of the the results, and so it may decide to use the sequential composition, that, exploiting the vicinity of N_3 and N_4, allows N_1 to obtain the output in the shortest time possible.

In this paper we exploit an algorithm (described in[6]) that is able to estimate the best alternative for providing a service to a seeker, exploiting services available on other mobile devices. The algorithm is based on a model that takes into account both the mobility of the devices and their computational capabilities in order to derive stochastic measures useful for comparing the possible alternatives to satisfy the service request. This algorithm selects the alternative that minimises the expected service execution time, defined as the time that would pass until the seeker receives the service results back when using a chosen alternative. It also takes into account whether the alternative is a single service or a composition, evaluating how the request resolution process may be impacted by disconnections between the nodes due to mobility.

The paper presents a set of simulation results obtained using the TheOne simulator[7], which has been proposed for evaluating opportunistic environments and includes different mobility models. We compare the performance of our algorithm against a set of alternative solutions. The rationale is considering lightweight solutions, that do not estimate service execution times as a function of the mobility of the nodes and load of the mobile cloud. We show that deriving and using these estimates pays off very significantly. In particular, our algorithm avoids saturating the resources of the mobile cloud by distributing the load much better. The resulting service execution time is far better than that obtained by the benchmarks in all conditions, and particularly when either the network starts becoming congested because of the traffic associated to the input/output data of service execution, or nodes becomes congested due to increased number of requests. We show that this holds true in all service composition cases, i.e. either when services are entirely available on individual nodes (no service composition), or when composition is the only possibility to provide the required function, or in mixed cases. Improvement of our algorithm with respect to the alternative solutions can be as high as 50, 40, 75% in these three cases, respectively, when network and provider saturation don't become unmanageable for the benchmark algorithms.

The paper is organized as follows. Section 2 describes the main approaches to service provisioning and composition in mobile environments. For the reader's convenience, we recall the main features of our algorithm for service selection and composition in section 3. Section 4 presents the performance results. Finally, concluding remarks are reported in section 5.

2. Related work

Research on mobile cloud computing shows a strong effort towards finding ways to relieve mobile devices from the execution of services, in favor of using the cloud infrastructure. In[8] authors present many different propositions for doing so, but in this approach the possibility of offloading computation or services execution to other mobile devices is rarely discussed. In[3] authors present cloud solutions where mobile devices provide services to other users, but such proposals consider mobility only as an exception. Conversely opportunistic computing handles mobility as a functional part of the system.

The goal of opportunistic computing is to allow the users to opportunistically compose the resources in the network. The sharing of resources in a heterogeneous mobile network can be used to compose functionalities not available in a single node of the network thus providing a much more rich functionality set. Such a vision requires new solutions for orchestration and management of resources on different devices[9]. Service composition exploits a mechanism based on graph theory that allows to collect the knowledge of the services offered and to represent this knowledge to evaluate the best alternative to use. Some recent research proposals take into accounts these aspects.[10] describes a system allowing service discovery and composition in networks with stable connectivity. The proposed system includes a mechanism for modelling services representing their semantics through the use of ontologies. They define a taxonomy of services through a list of concepts describing them[11]. A key problem of service provisioning through direct communication between nodes in mobile environments concerns the possibility that possible service providers may not be directly reachable (even through a multi-hop ad hoc path) when the seeker issues a request.

In opportunistic computing, instead, the fact that nodes may not be reachable is considered as the rule, and proposed mechanisms are designed correspondingly. For example, in[12] fault tolerance is achieved through a middleware that exploits information about the devices to create parallel service compositions, in order to increase the probability for successful executions.[13] proposes some heuristics for service composition taking into account the last time of encounter between nodes and the reported load at providers.

In[14, 15] the problem is addressed using a stochastic analysis of the providers in order to find the ones that are most likely reachable from the seeker during the entire period necessary to perform the composition. In these works, the loss of connection is considered as a failure state.

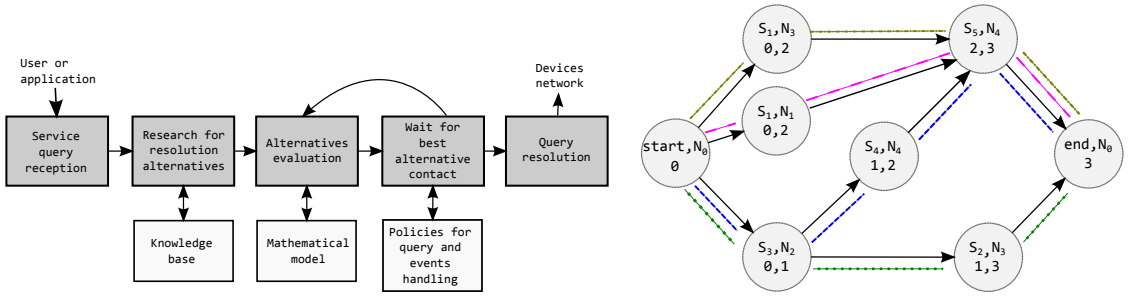
Authors of[16] analyse the issue of single service provision in opportunistic networks. The main aim is to improve the efficiency of service provision by replicating requests to a set of different providers. The proper number of requests is computed by considering information on the mobility of the users and on the load of the nodes. The optimal number of requests is computed through an analytical model that minimizes the expected value of the request resolution time, defined as the time required to receive the first response from suppliers.

Finally, in[6] we have proposed an algorithm extending previous techniques whose aim is to minimize the time for

the provisioning of a composite service. With respect to [12] and [13] this is based on an analytical model for computing the expected completion time of each alternative, rather than on simple heuristics for selecting the composition (similar in design to the ones we compare against in simulations). With respect to [16], the algorithm presented in [6] considers the possibility of composing different services.

3. Service selection and composition algorithm

For the reader's convenience, in this section we briefly summarise the main features of the service provisioning algorithm proposed in [6]. This algorithm enables either a user or an application of a mobile network to compose local and remote services to satisfy a user request. Even if a service request may be satisfied either by an atomic service or by a composition of services, in the following we will describe the more general scenario of service composition. To describe the system behaviour in more details, we show how a service request is managed and which components are involved in the resolution of the request. Consider Figure 2a. When a node generates a request, it searches for



(a) Service request resolution

(b) The composition graph

Figure 2: System mechanics and composition evaluation

alternative service compositions satisfying it. To this end, the node keeps a graph with its knowledge about nodes, and the services they provide. A vertex in the graph represents a service and the node that provides it, along with the abstract representation of the service input and output data types, so that edges are created considering the type compatibilities between the services. A weight is associated to each edge, representing the cost (i.e., the expected time) to execute the service on the node corresponding to the vertex at the endpoint of the edge (the graph is directed). The graph is built by exploiting the information present in the knowledge base about the services available in the network (to the best of the node's knowledge) and the providers offering them, as seen in Figure 2b. The weights paired with the arcs of the graph are dynamically updated when fresh information is collected upon direct contacts between nodes. The weights are computed based on estimates of the times to complete the different phases required to provide the service at the corresponding node, as described in Section 3.1.

Once the graph has been weighed, a shortest-path algorithm is used to choose, among the alternative compositions, the one providing the best expected service provisioning time (i.e., the total time the seeker has to wait to obtain the service results). In our system we assume that a service execution (or a component of a composed service) can be requested only when the seeker and the provider are in direct contact (in case of composition, when the provider of the previous component is in direct contact with the provider of the next component). Once the best alternative is detected, if the chosen provider for the first component cannot be directly contacted, the node waits for a contact with it. While waiting for this, it may encounter further nodes and collect new information from them. Since this information may alter the classification computed previously, the compositions must be re-evaluated whenever fresh information is received from the opportunistic contacts before contacting the first provider of the best path.

The task of the events handling component is to keep monitoring both the network and the node itself to catch events that may affect either the resolution of a pending request or, in general, the local knowledge base. The event handling component triggers either the transmission of fresh information on the network as a response to an event generated by the local node, or the refreshing of the local knowledge base as a response to the reception of new information received from the network.

3.1. Estimation of service provisioning time

In this section we sketch the way how service provisioning time is estimated, considering the general case where a service needs to be composed out of a certain number of components. In other words, this is how a seeker estimates the time required by the different service composition alternatives available based on the graph representation given in Figure 2b. For a given service composition alternative, we define R as a random variable for the time needed to complete the request resolution process, whose expected value will be exploited to choose the best alternative. R is defined in terms of a set of random variables corresponding to the different phases and the length n of the evaluated composition:

$$R = W + B + \sum_{i=1}^{n} (DQ_i + DS_i + \theta_i)$$

W is the time needed for the seeker to contact the first service provider that is determined by the intercontact time between the seeker and the provider. This value depends on their relative mobility. For each pair of nodes, we assume that contact and intercontact times follow exponential distributions and are independent and identically distributed (i.i.d.). We also assume that contact and intercontact times of different pairs of nodes are independent of each other.

B and θ_i are respectively the time needed to transfer the input data for the service composition from the seeker to the first provider, and the time needed to trasfer the results of the i-th service execution to the next node involved in the composition (this may also include the time to encounter the node after the end of the execution of the previous component). Note that in an opportunistic network data transfers between two nodes may be affected by connection disruptions, due to the nodes mobility, but also by transfer from and to other nodes that use the same shared medium, or even by other concurrent transfers between the seeker and the provider involving other requests or activities. This implies that, in general, both the duration of contact and intercontact times impact on the duration of data transfers.

DQ_i is the time spent by the request in the queue of the i-th service provider due to previous pending executions (we model this as a FIFO queue at the provider). The duration of this delay depends both on the frequency of the request arrivals to the provider and on the time to process them. To model it, we consider the $M^{[X]}/G/1$ [17] queueing model, where nodes generate requests according to a Poisson distribution and send batches of requests to the provider (upon encountering). Finally, DS_i is the time to execute the i-th service of the composition, that depends both on its computational capabilities and on the type of the service. Details for the derivation of closed form expressions for these variables are available in in [6].

4. System Evaluation

In this section we analyse the behaviour of our system in different simulated scenarios in order to evaluate its effectiveness against other policies.

For this purpose, we developed a set of simulations through TheOne [7]. The experiments exploit a set of mobility traces generated according to the RandomWayPoint model, modified as discussed in [18] in order to avoid problems related to the initial transient phase of the mobility model. Note that, although in general other mobility models are considered more realistic, RWP is still a valid option when users form a unique social community moving in a common area [19].

For each simulation scenario, we averaged the results upon 5 simulation runs. We consider 30 devices that move in a square with sides $500m$ long with the devices having a $90m$ transmission range and a 2 $Mbps$ maximum bandwidth, so there is a high probability that each node can have many different direct contacts with other nodes at any time during the simulation. With this scenario we could test our system when choosing between many different suitable alternatives.

Each simulation run covers $400000s$ of simulated time, with the first $10000s$ allotted to only the exchange of mobility data to build the knowledge base of the nodes. Then service requests are generated for $360000s$ (100 simulated hours). The last $30000s$ are left to complete the service resolution processes.

Each service is identified by the type of its input/output which is codified by an integer. In our simulation an input type i is selected in the integer range [0,7], while output type o in the range [1,8], with the constraint that i is less than o to avoid cyclic compositions. Each service can be provided by 25% of the nodes of the network. To evaluate the

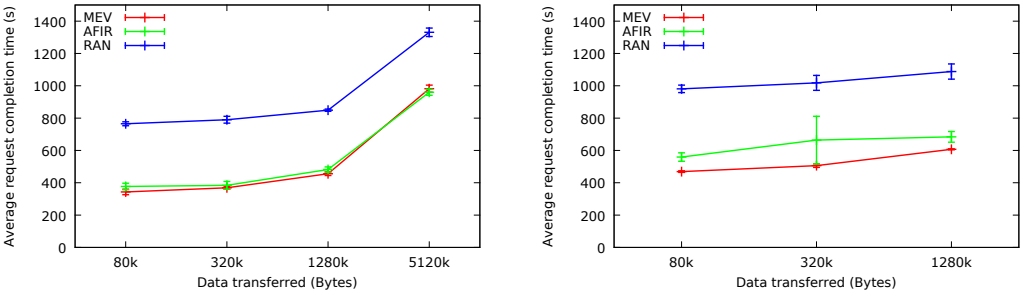
performance of our system, we pit our system against other service selection policies which are simpler to implement, but do not estimate service provisioning times. The comparison is done by evaluating, for each policy, its average request completion time in different scenarios.

The *Minimum Expected Value (MEV)* policy selects service compositions according to our algorithm. The *Random (RAN)* policy selects for each request, a random path in the graph of compositions. In the *Always First (AFIR)* policy, the seeker chooses, as the first provider, a random suitable node that has a contact already established and that can progress the resolution process towards the final output. If there are no such contacts, it waits until it encounters such a provider. This evaluation is repeated by each provider that receives a service request after their service has terminated its execution, until the final output is obtained and transferred from the last provider to the seeker. Finally, in the *Atomic (ATOM)* policy the seeker waits for a provider that offers a single service satisfying the request. For this policy compositions are not taken into account at all.

The behaviour of these policies are evaluated in three main scenarios. In the first one, all the requests are evaluated and solved using exclusively single service executions (we will call this scenario "Single"), so that compositions cannot be used (and the RAN and ATOM policies coincide). In the second one (the "Comp" scenario) requests cannot be solved using a single service (ATOM is not evaluated in this scenario because of the inability to abide to the scenario limitation). Finally, we consider a "Mixed" scenario, where all four policies can use any composition length, according to the policy behaviour.

For all scenarios, we have run two sets of simulations where different amounts of requests are generated. In the "5-8" scenario, a new request is generated after the previous one, after a time uniformly distributed in the interval between 5 and 8 seconds has passed. Each request is then assigned to a randomly chosen node, also following a uniform distribution, who will act as a seeker. The "20-40" scenario is similar to "5-8", but the intervals between request generations are chosen between 20 and 40 seconds.

Finally, we consider different sizes for the input/output parameters of the service components: 80 KBytes, 320 KBytes and 1280 KBytes for the "5-8" setting and 80 KBytes, 320KBytes, 1280 KBytes and 5120 KBytes for the "20-40" setting. In each simulation, the input and output data sizes are the same for all services and requests. All the results shown are the average results of the 5 independent simulation runs executed for each scenario. We also show the confidence intervals, with a 95% confidence level.



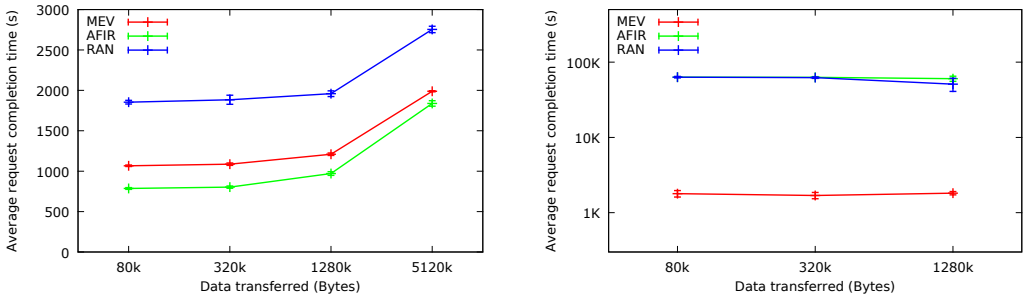
(a) load 20-40 (b) load 5-8

Figure 3: Request completion time by data transfer sizes, scenario "Single"

In the "Single" scenario we force each policy to use the least possible number of service requests to solve any generated user request. In Figure 3a we can see the average request completion time for various data transfer size settings with a low request load (inter-generation interval "20-40"). AFIR and MEV show similar performance, and both drastically outperform RAN, with up to 50% less time spent on small input data sizes.

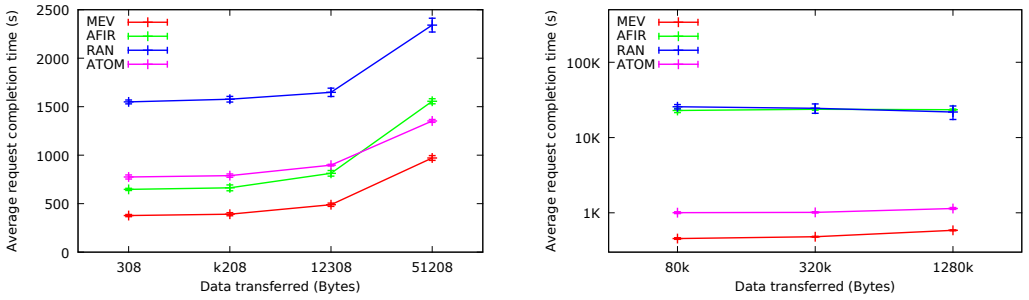
Once we also set a higher load of requests to the system, we can see (in Figure 3b) that the advantage in using MEV over AFIR becomes more apparent, with MEV coping better with the increase of the queue waiting time, keeping at least at 45% the time saved against RAN.

In the "Comp" scenario, the unavailability of single service solutions favours the use of solutions that can handle the load of requests without flooding the network or that can save time on each data transfer phase in the compositions.



(a) load 20-40 (b) load 5-8

Figure 4: Request completion time by data transfer sizes, scenario “Comp”



(a) load 20-40 (b) load 5-8

Figure 5: Request completion time by data transfer sizes, scenario “Mixed”

We can see in Figure 4a that with low request load, AFIR manages to outperform MEV exploiting the knowledge of the state of the connections between providers after each service is executed. This advantage is amplified by low data transfer sizes that make possible to exchange data before the used contact may end. This is confirmed noting that with bigger data sizes, the difference between MEV and AFIR decreases. When compared against RAN, MEV still manages to achieve up to 40% better performances. With an heavier load scenario (as seen in Figure 4b), both AFIR and RAN cannot handle the flow of requests given by the use of compositions, with average times 40 times greater than the ones achieved by MEV (note the logarithmic scale on the y axis).

In the “Mixed” scenario, each policy has the opportunity to use any possible composition length to solve a request (apart from ATOM that uses only single service executions). In this kind of scenario, MEV can exploit the wide range of choices keeping into account the stability of the network. In Figure 5a, we can see how MEV outperforms the other policies even at low loads, saving near to 40% of time in respect with AFIR and 75% with respect to RAN with low transfer data sizes. With bigger data sizes, MEV outperforms ATOM with at least 35% shorter times, exploiting the fact that ATOM isn’t aware of the computing capabilities of the providers.

In Figure 5b we see the high load scenario, where the difference between MEV and the other policies is even more remarkable, with AFIR and RAN suffering from overloading the nodes, and ATOM being incapable of giving good performances generating the least possible amount of requests between all policies, with MEV achieving times near to 50% shorter (note again the logarithmic scale in the y axis).

The under-performing of AFIR is caused by its choices of compositions for solving requests. Without any guidance about the load state and the service execution performance of each provider in the composition paths, AFIR is not able to exploit the knowledge of the state of the network at each execution step. Also, without any reference about the lengths of the chosen compositions, AFIR tends to saturate the provider with service requests.

5. Conclusions

In this paper we have evaluated a system able to select and compose services for a mobile cloud environment through the opportunistic computing environment. The system is based on a mathematical model to evaluate alternatives known by the seeker when the request is generated, taking into account the characteristics of the compositions and the status of the network and of possible nodes providing the service. The simulation results show that our system outperforms other policies in a varied range of scenarios in terms of average resolution times of users' requests. In particular, our solution is drastically better as the network becomes more and more congested, either due to load at service providers or congestion at the network level. This is because our solution is able to detect those condition, spreads the load more intelligently, and thus avoids creating bottlenecks in the service provisioning process.

Acknowledgments

This work was partially funded by the European Commission under the MOTO (FP7 317959), FIRE EINS (FP7-288021), and EIT ICT Labs MOSES (Business Plan 2014) projects.

References

1. Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018", Feb. 2014.
2. N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey", *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84-106, Jan. 2013.
3. S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-Based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges," *Communications Surveys & Tutorials, IEEE*, vol.16, no.1, pp.337-368, 2014.
4. L. Pelusi, A. Passarella, and M. Conti, "Opportunistic networking: data forwarding in disconnected mobile ad hoc networks," *IEEE Commun. Mag.*, vol. 44, no. 11, pp. 134–141, Nov. 2006.
5. M. Conti and M. Kumar, "Opportunities in opportunistic computing," *Computer*, vol. 43, no. 1, pp. 42–50, Jan. 2010.
6. M. Conti, E. Marzini, D. Mascitti, A. Passarella and L. Ricci, "Service selection and composition in opportunistic networks," *Prooceedings of the 9th International Wireless Communications and Mobile Computing Conference (IWCMC),* pp. 1565 – 1572, 2013.
7. A. Kernen, J. Ott, and T. Krkkinen, "The ONE simulator for DTN protocol evaluation," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, ser. Simutools '09, 2009, pp. 55:1–55:10.
8. K. Kumar, J. Liu, Y. Lu, and B. Bhargava, "A Survey of Computation Offloading for Mobile Systems", *Mob. Netw. Appl.* vol. 18, no. 1, pp. 129-140, Feb. 2013.
9. M. Conti, S. Giordano, M. May, and A. Passarella, "From opportunistic networks to opportunistic computing," *IEEE Commun. Mag.*, vol. 48, no. 9, pp. 126–139, Sep. 2010.
10. S. Kalasapur, M. Kumar, and B. A. Shirazi, "Dynamic service composition in pervasive computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 7, pp. 907–918, Jul. 2007.
11. D. Bianchini, V. D. Antonellis, and M. Melchiori, "An ontology-based architecture for service discovery and advice system," in *Database and Expert Systems Applications, Proceedings. Sixteenth International Workshop on*, Aug. 2005, pp. 551–556.
12. S. A. Tamhane, M. Kumar, A. Passarella, and M. Conti, "Service composition in opportunistic networks," in *Green Computing and Communications (GreenCom), IEEE International Conference on*, Nov. 2012, pp. 285–292.
13. U. Sadiq, M. Kumar, A. Passarella, and M. Conti, "Modeling and simulation of service composition in opportunistic networks," in *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, ser. MSWiM '11, 2011, pp. 159–168.
14. L. D. Prete and L. Capra, "Reliable discovery and selection of composite services in mobile environments," in *Enterprise Distributed Object Computing Conference, EDOC '08. 12th International IEEE*, Sep. 2008, pp. 171–180.
15. J. Wang, "Exploiting mobility prediction for dependable service composition in wireless mobile ad hoc networks," *IEEE Trans. Serv. Comput.*, vol. 4, no. 1, pp. 44–55, Jan. 2011.
16. A. Passarella, M. Kumar, M. Conti, and E. Borgia, "Minimum-delay service provisioning in opportunistic networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 8, pp. 1267–1275, Aug. 2011.
17. H. Takagi, *Vacation and Priority Systems, Part 1*. Elsevier Science Publishers B.V., 1991.
18. W. Navidi and T. Camp, "Stationary distributions for the random waypoint mobility model," *IEEE Trans. Mobile Comput.*, vol. 3, no. 1, pp. 99–108, Jan. 2004.
19. C. Boldrini and A. Passarella, "HCMM: Modelling spatial and temporal properties of human mobility driven by users social relationships," *Computer Communications*, vol. 33, no. 9, pp. 1056 – 1074, 2010.